**Please answer these questions for Big Data Engineer position**

**Skill Matrix:**

|  |  |
| --- | --- |
| **Technical Acumen** | **Rating out of 10** |
| Strong programming experience either in Python or Java or Scala | 8.5 |
| Strong SQL experience in writing and analysing complex and lengthy queries | 7 |
| Strong experience in writing Spark jobs and Strong understanding of Spark architecture | 8 |
| Pyspark | 8 |

Scala

1) What is the super class of all classes in Scala?

Scala.any is the super class of all the classes in scala.

2) Why do we use Tuples in Scala?

Tuples contain a fixed number of elements and are immutable. This allows for bundling data and ensuring they are thread-safe allowing for parallel processing.

3) What are different Collection data types in Scala?

Collections in scala can have to types mutable and immutable. Here are some collection data types: Traversable, iterable, seq, set, map, index & linear sequence, sorted set & map, bitset.

1) Is Spark faster than Map-Reduce and why?

Spark is faster than Map-Reduce (MR) because Map-reduce is designed to run on any (commodity) hardware configurationg the MR jobs are using slow disk/ssd storage compared to Spark that uses RAM memory which is orders of magnitude faster than ssd/hdd hardware.

2) What are actions and transformations?

Transformation are operations performed on an RDD to create a new rdd.

Actions apply the computation to the transformation and return the result to the driver.

3) What is the difference between cache () and persist () methods

Cache method will store the RDD/Dataframe in memory and will not be re- read from storage. The persist method can store the information in multiple configuration to be suit the job: MEM ONLY, MEM AND STORAGE, STORAGE ONLY, and other job specific options.

4) What is Lazy evaluation in Spark.

Lazy evaluation is the opposite of eager evaluation where evaluation is done as soon as possible (bound to a variable). Lazy evaluation is used in spark to optimize the rdd lineage by optimizing the the Directed Acyclic Graph (DAG) which holds the transformations that get called when an action is performed. This has a positive impact by saving computation time and increased speed of the spark job.

5) Spark executors are calculated based on the number of cores available in cluster.

total\_cores = (cores\_per\_node -1) \* nodes\_in\_cluster

executors = (total\_cores/num\_cores\_per\_executor) -1 #App manager

We want to keep 1 core for yarn/hadoop daemons and 1 executor for application manager.

Data Modelling

1) What are SCD Type 2 Tables?

Type 2 Slow Changing dimension tables will track changes to records by adding timestamp field. A way to track start and end dates. Effective dates with a current flag.

2) What are SCD Type 1 Tables?

Type 1 Slow changing dimension tables overwrite the original data and the record will not have any historical information.

3) What is fact less fact tables?

Fact less tables are tables that have no measures (data) but have only dimensional keys (foreign keys) which describe the condition/time period fact.

4) What is a conﬁrmed Dimension?

A conformed dimension can refer to multiple tables in multiple data marts within the same organization. It’s a way to connect mutiple tables with common data with other tables without creating duplicate tables like a time dimension.

Problem 1:

Compress String Problem

Problem Description: Given a string, return a compressed string with number of times a character is repeated only if the character is repeated more than three times consecutively.

Input1: uuuuuuuuuuuu Output1: 12Xu

Input2: abbccccuiiiiii Output2: abb4Xcu6Xi

PYTHON 3

**def** string\_problem(string\_iniput):

*'''*

*This method will map the string input value and associated count*

*for the numbers of times a character is used.*

**:param** *string\_iniput: This is an input string that you need to map for the frequency*

*of the characters used.*

**:return***: dictionary of keys used A-Z*

*'''*

keys = {}

**for** chars **in** string\_iniput:

*# Increment the keys value for duplicates. I.e 'aaa' map to 'a:3'*

**if** chars **in** keys **and** keys[chars]:

keys[chars] += 1

**else**:

keys[chars] = 1

**return** keys

**def** string\_encode(dict\_keys):

*'''*

*This method will encode the keys based on the dictionary keys passed to it.*

**:param** *dict\_keys: Keys to be encoded (compressed)*

**:return***: A compressed output of the keys*

*'''*

output = **""**

**for** key **in** dict\_keys:

**if** dict\_keys[key] > 3:

output += str(dict\_keys[key]) + **"X"** + key

**else**:

output += key \* dict\_keys[key]

**return** output

**def** main(value=**"uuuuuuuuuuuu"**, value2=**"abbccccuiiiiii"**):

*'''*

*This class will compress string with number of times a character*

*is repeated only if the character is repeated more than three times consecutively.*

*The output will be displayed in terminal/console.*

**:return***: None*

*'''*

value = **"uuuuuuuuuuuu"**

value2 = **"abbccccuiiiiii"**

key\_value1 = string\_problem(value)

key\_value2 = string\_problem(value2)

encode\_keys = string\_encode(key\_value1)

encode\_keys2 = string\_encode(key\_value2)

print(encode\_keys)

print(encode\_keys2)

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

main()

Problem 2:

Sort String Using Given Function Problem Description: Given a string and given function which return only true or false, return a sorted string using only the given function.

Given functions:

public Boolean is A( ) { return true }

public Boolean isB()

{ return true }

public Boolean isC()

{ return true }

Input String: AABCABBA

Expected Output: AAAABBBC

PYTHON 3:

**def** isA(input):

**return True if** input == **"A" else False**

**def** isB(input):

**return True if** input == **"B" else False**

**def** isC(input):

**return True if** input == **"C" else False**

test\_case = [**"A"**,**"B"**,**"C"**]

**for** x **in** test\_case:

print(isA(x),isB(x),isC(x))

input\_string1 = **"AABCABBA"**

output = **""**

**for** x **in** input\_string1:

**if** isA(x) **and not** isB(x) **and not** isC(x):

output += x

**for** x **in** input\_string1:

**if not** isA(x) **and** isB(x) **and not** isC(x):

output += x

**for** x **in** input\_string1:

**if not** isA(x) **and not** isB(x) **and** isC(x):

output += x

print(**"\nMy output: "**+output)